uc3m | Universidad Carlos III de Madrid

Tema 1 - Tipos Abstractos de Datos

Hoja de Problemas

Problema 1 - Implementa una clase de Python que nos permita representar tarjetas de crédito para clientes de banco. Cada tarjeta de crédito debe contener la siguiente información:

- el nombre del cliente
- un identificador de la tarjeta
- el límite de la tarjeta, es decir, la cantidad total que el cliente puede gastar cada mes.
- el balance de la tarjeta, es decir, la cantidad de dinero gastado.

Además, las siguientes operaciones deben ser consideradas:

- una operación para hacer un cargo en la tarjeta. Es decir, esta operación recibe una cantidad de dinero e incremente el balance de la tarjeta. Si el nuevo balance excede el límite de la tarjeta de crédito, la operación no debe ser permitida.
- una operación que permita hace un depósito en la tarjeta. Es decir, esta operación recibe una cantidad de dinero y disminuye el balance. El depósito nunca debe exceder el actua balance de la tarjeta.

Nota: Por favor, sigue las especificaciones mencionadas anteriormente. No hagas suposiciones diferentes sobre cómo deberían funcionar las tarjeta de crédito. Este enunciado es simplemente un ejemplo para práctica la implementación de una estructura de datos en Python.

Problema 2: Implementa una clase, **Vector**, para representar vectores en un espacio multidimensional. Por ejemplo:

En un espacio de 3 dimensiones, podríamos querer representar un vector con las siguientes coordenadas: 5, -2, 3 .

En un espacio de 3 dimensiones, podríamos querer representar un vector con las siguientes coordenadas: 0,1,-1,3,2.

La clase debe contener los siguientes métodos:

- __init__(**self**,dim): método constructor que crea un vector con todas las coordenadas igual a 0 y de dimensión m.
- __len__(**self**): devuelve la dimensión del vector.
- __str__(**self**): devuelve un string representando el vector. Por ejemplo, si las coordenadas del vector son: 3,5,0, el método debería devolver la cadena: "(3,5,0)".
- __getitem__(**self**,i): devuelve la i ésima coordenada del vector. the ith coordinate of the vector. Recuerda que la primera coordenada siempre debe estar representada por el índice 0.
- __setitem__(**self**,i,newValue): modifica la i ésima coordenada del vector al nuevo valor newValue.
- __add__(**self**,other): devuelve un nuevo vector que es la suma del vector invocante (self) y del parámetro other.
- __eq__(**self**,other): devuelve True is los dos vectores son iguales, y False en otro caso.
- dot(**self**,other): devuelve el producto escalar de dos vectores.
- cosine_distance(**self**,other): devuelve la distancia del coseno entre los dos vectores.

Problema 3: Un polinomio es una expresión algebraica, $Q(x)=a_0+a_1x+a_2x^2+a_3x^3+...+a_nx^n$, formada por un monomio o por la suma de varios monomios. Un monomio es una expresión algebraica formada por un coeficiente, una variable (generalmente x) y un exponente, por ejemplo: $5x^3$.

Implementa una clase, Polynomial, que permita representar un polinomio. Las operaciones que :

- Un método constructor que reciba como argumento de entrada un array (lista de Python) con los coeficientes del polinomio. El elemento en el índice 0 se corresponde con el coeficiente del término de grado 0 (es decir, es el término constane). El elemento almacenado en el índice 1 corresponde al coeficiente del término con grado 1, y así sucesivamente.
- Un método que devuelva el grado del polinomio. Por ejemplo, Q(x)=5 tiene grado 0. $Q(x)=x^2+5$ tiene grado 2.
- Un método que reciba un entero positivo, n, y devuelva el coeficiente del término elevado a n. Por ejemplo, dado el polinomio, $Q(x)=a_0+a_1x+a_2x^2+a_3x^3+...+a_nx^n$, si n=3, el coeficiente es a_3.
- Un método que reciba un entero, n, y un valor, new_value, y modifique el coeficiente del término n por el nuevo valor, new_value. Por ejemplo, dado el polinomio $Q(x)=a_0+a_1x+a_2x^2+a_3x^3+...+a_nx^n$, para n=3, y new_value=b, el polinomio quedaría modificado a $Q(x)=a_0+a_1x+a_2x^2+bx^3+...+a_nx^n$

- Un método que reciba un entero, x, y devuelva el valor del polinomio para ese valor. Por ejemplo, para $Q(x)=3x^2+4x+5$, $Q(1)=12$.
- Un método que reciba un objeto de tipo Polynomial, p, y que devuelva un nuevo objeto Polynomial que sea resultado de sumar el objeto invocante y el polinomio p. Por ejemplo, $Q(x)=3x^2+4x+5$, $p(x)=x^3+4x^2-2x-3$. Q.sum(p) = x^3+7x^2+2x+2 .

Problema 3: Python 3 ya incluye una función, **range**, que permite obtener secuencias de enteros. Vamos a implementar nuestra propia clase **Range** que permita almacenar una secuencia de enteros. Nuestra clase contiene los siguientes atributos:

- **start**: valor inicial de la secuencia.
- **end**: límite superior de la secuencia. No pertenece a la secuencia.
- **step**: distancia entre los números de la secuencia. No puede ser 0.

Por ejemplo, r=Range(2, 10, 2), r contendrá los siguientes valores [2, 4, 6, 8].

Los métodos de la clase son:

- un método constructor que toma tres parámetros para inicializar los atributos start, end y step. El valor por defecto de step es 1.
- un método que devuelva el número de elementos en el rango Por ejemplo, para r = Range(2, 10, 2), su tamaño es 4.
- un método que reciba un valor entero, i>=0, y devuelve el i-ésimo elemento en el rango. Por ejemplo, r[0]=2,r[1]=4,r[2]=6,r[3]=8.
- un método que devuelva una cadena que contenga la secuencia de elementos en este rango**.** Por ejemplo, str(r)='2,4,6,8'
- un método que devuelva la suma de todos los elementos de la secuencia**.** sum(r)=2+4+6+8=20

Incluye instrucciones para testear cada método.

Nota: Recuerda que los índices para las colecciones comienzan por 0 en la mayoría de los lenguajes de Programación (Python, Java, C, C++). Es decir, el primer elemento de la colección tiene un índice 0, mientras el último elemento tiene un índice de longitud menos uno. No puedes usar la función rane de Python en tu implementación.